TPCC-UVa

An Open-Source TPC-C Implementation for Parallel and Distributed Systems

Diego R. Llanos, Belén Palop

Universidad de Valladolid

Computer Science Department
University of Valladolid, Spain

PMEO-PDS 06, Rhodes Island, April 29th, 2006

Motivation

- There are many benchmarks available to measure CPU performance:
  - SPEC CPU2000, NAS, Olden...
- To measure global system performance, vendors use TPC-C benchmark
- However, only TPC-C specifications are freely available
- TPCC-UVa is an (unofficial) implementation of the TPC-C benchmark, intended for research purposes

How does TPC-C work?

- TPC-C simulates the execution of a set of both interactive and deferred transactions: OLTP-like environment
- A number of terminals request the execution of different database transactions, simulating a wholesale supplier
- Five different transaction types are executed during a 2- to 8-hours period:
  - **New Order** enters a complete order
  - **Payment** enters the customer's payment
  - **Order Status** queries the status of a customer's last order
  - **Delivery** processes a batch of ten new orders
  - **Stock Level** determines the number of recently sold items
- The number of **New Order** transactions processed during the measurement time gives the performance number: Transactions-per-minute-C, or tpm-C

How does TPC-C work?

- TPC-C simulates the execution of a set of both interactive and deferred transactions: OLTP-like environment
- A number of terminals request the execution of different database transactions, simulating a wholesale supplier
- Five different transaction types are executed during a 2- to 8-hours period:
  - **New Order** enters a complete order
  - **Payment** enters the customer's payment
  - **Order Status** queries the status of a customer's last order
  - **Delivery** processes a batch of ten new orders
  - **Stock Level** determines the number of recently sold items
- The number of **New Order** transactions processed during the measurement time gives the performance number: Transactions-per-minute-C, or tpm-C

How does TPC-C work?

- TPC-C simulates the execution of a set of both interactive and deferred transactions: OLTP-like environment
- A number of terminals request the execution of different database transactions, simulating a wholesale supplier
- Five different transaction types are executed during a 2- to 8-hours period:
  - **New Order** enters a complete order
  - **Payment** enters the customer's payment
  - **Order Status** queries the status of a customer's last order
  - **Delivery** processes a batch of ten new orders
  - **Stock Level** determines the number of recently sold items
- The number of **New Order** transactions processed during the measurement time gives the performance number: Transactions-per-minute-C, or tpm-C

How does TPC-C work?

- TPC-C simulates the execution of a set of both interactive and deferred transactions: OLTP-like environment
- A number of terminals request the execution of different database transactions, simulating a wholesale supplier
- Five different transaction types are executed during a 2- to 8-hours period:
  - **New Order** enters a complete order
  - **Payment** enters the customer's payment
  - **Order Status** queries the status of a customer's last order
  - **Delivery** processes a batch of ten new orders
  - **Stock Level** determines the number of recently sold items
- The number of **New Order** transactions processed during the measurement time gives the performance number: Transactions-per-minute-C, or tpm-C

About TPCC-UVa

Disclaimer

TPCC-UVa is *not* an official implementation. Our performance number, tpmC-uva, should not be compared with tpm-C given by any vendor

- Why not?
  - We have not implemented price-per-tpmC metrics
  - Our Transaction Monitor is not “commercially available”
  - Therefore, the implementation does not have TPC approval
- TPCC-UVa is written entirely in C language, and uses the PostgreSQL database engine
- To ensure fairness, we distribute TPCC-UVa together with the toolchain that should be used to compile it

About TPCC-UVa

Disclaimer

TPCC-UVa is *not* an official implementation. Our performance number, tpmC-uva, should not be compared with tpm-C given by any vendor

- Why not?
  - We have not implemented price-per-tpmC metrics
  - Our Transaction Monitor is not “commercially available”
  - Therefore, the implementation does not have TPC approval
- TPCC-UVa is written entirely in C language, and uses the PostgreSQL database engine
- To ensure fairness, we distribute TPCC-UVa together with the toolchain that should be used to compile it

About TPCC-UVa

Disclaimer

TPCC-UVa is *not* an official implementation. Our performance number, tpmC-uva, should not be compared with tpm-C given by any vendor

- Why not?
  - We have not implemented price-per-tpmC metrics
  - Our Transaction Monitor is not “commercially available”
  - Therefore, the implementation does not have TPC approval
- TPCC-UVa is written entirely in C language, and uses the PostgreSQL database engine
- To ensure fairness, we distribute TPCC-UVa together with the toolchain that should be used to compile it

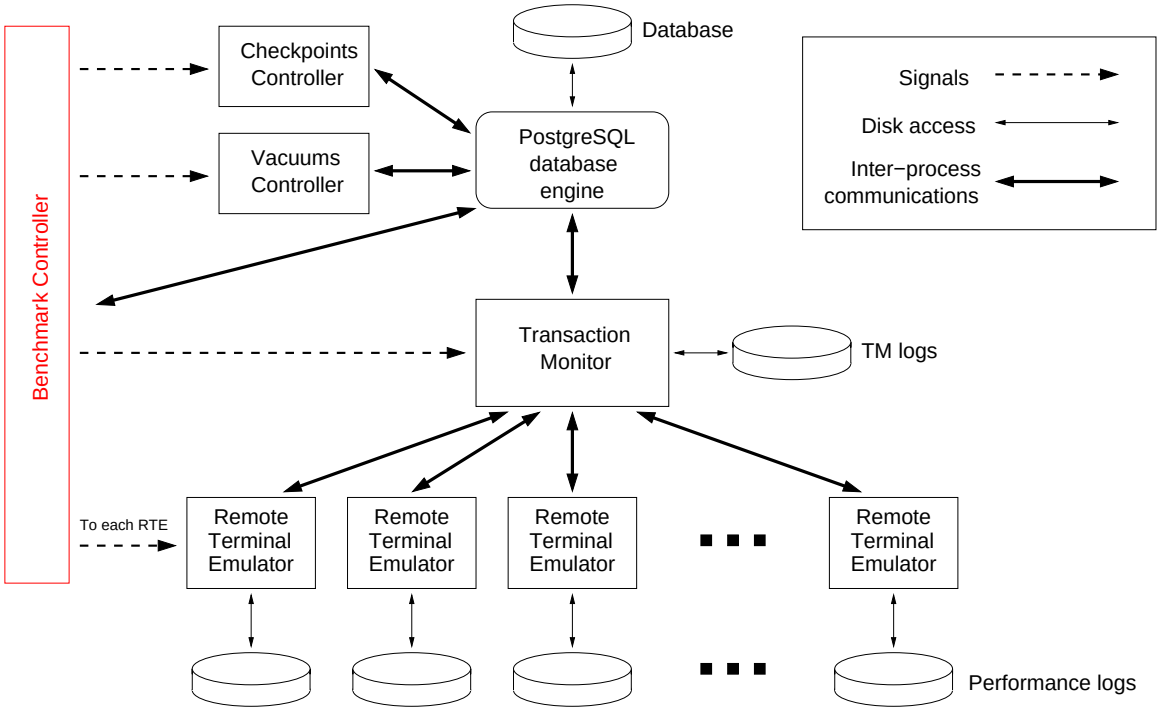
About TPCC-UVa

Disclaimer

TPCC-UVa is *not* an official implementation. Our performance number, tpmC-uva, should not be compared with tpm-C given by any vendor

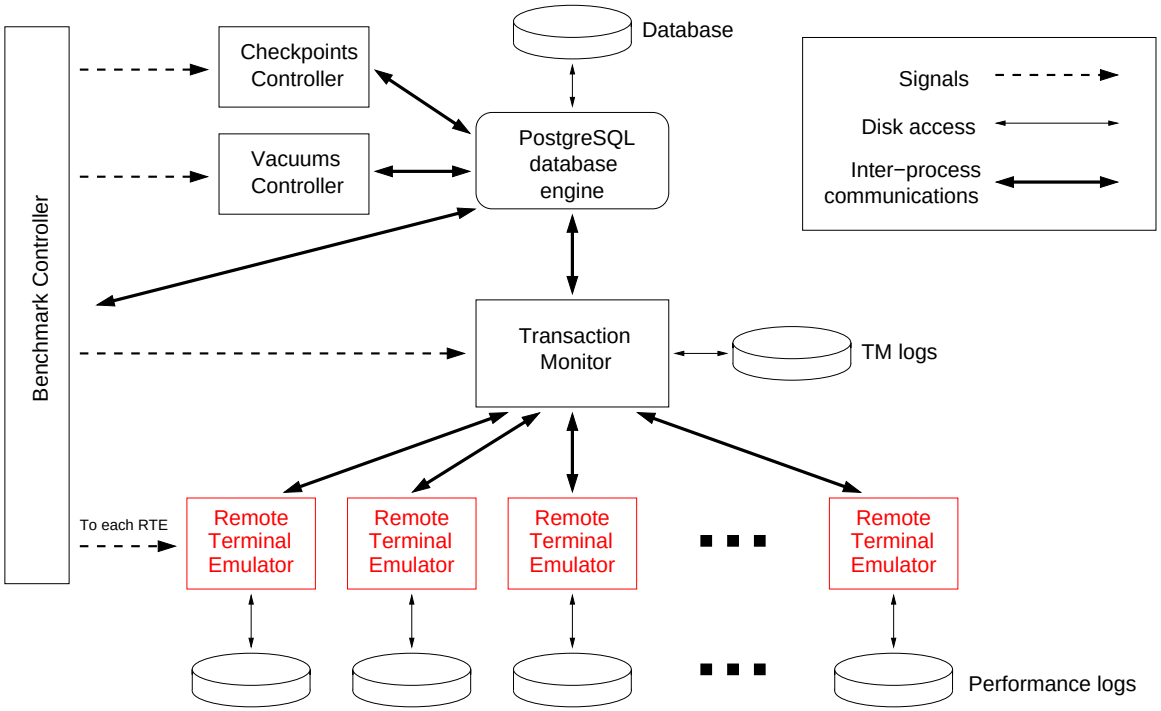
- Why not?
  - We have not implemented price-per-tpmC metrics
  - Our Transaction Monitor is not “commercially available”
  - Therefore, the implementation does not have TPC approval
- TPCC-UVa is written entirely in C language, and uses the PostgreSQL database engine
- To ensure fairness, we distribute TPCC-UVa together with the toolchain that should be used to compile it

TPCC-UVa architecture



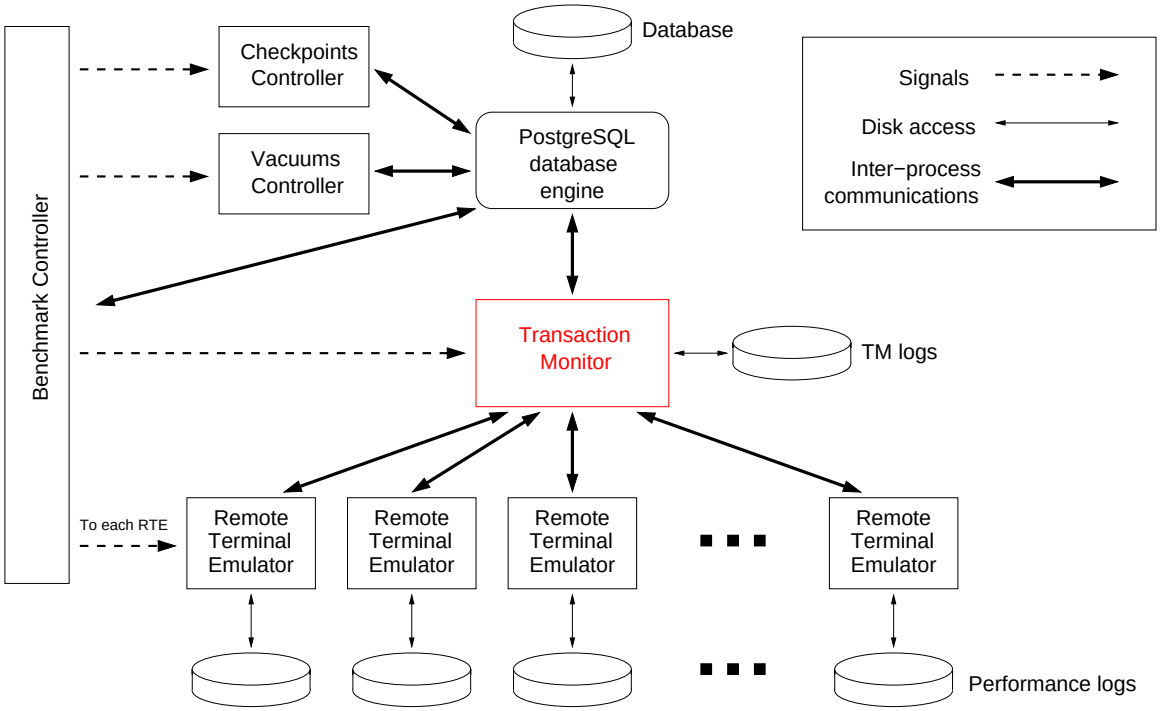
- The Benchmark Controller interacts with the user, populating database and launching experiments

TPCC-UVa architecture



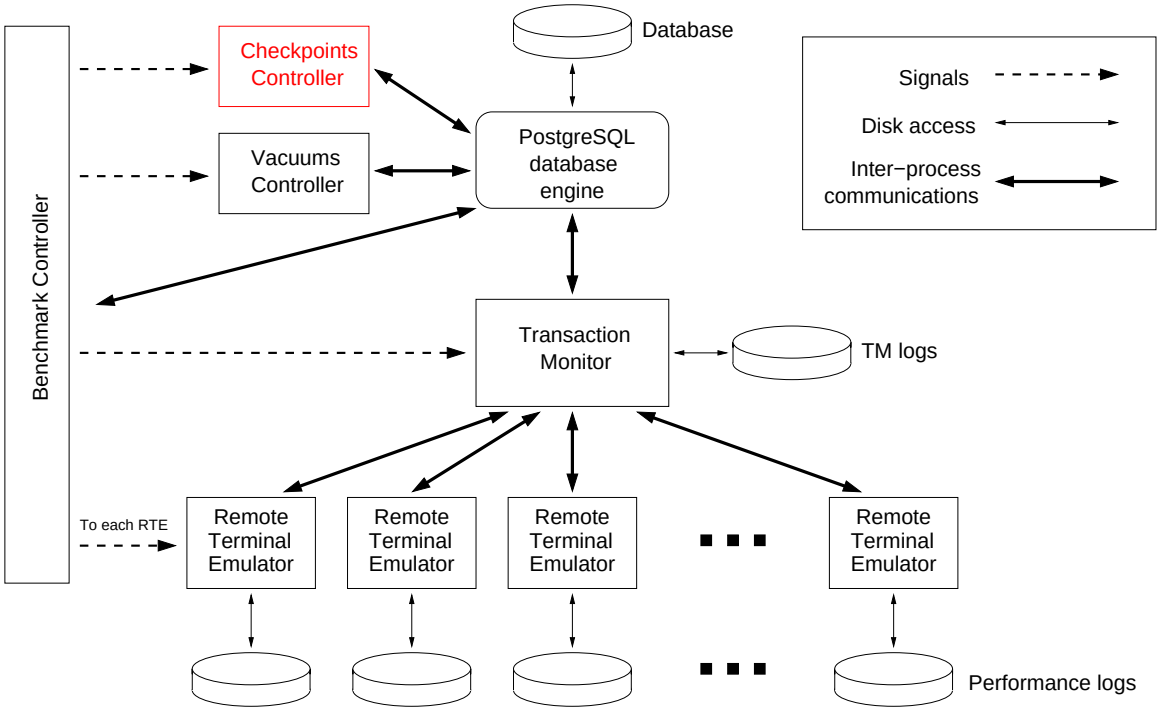
- The Remote Terminal Emulators, one por terminal, request transactions according with TPC-C specifications

TPCC-UVa architecture



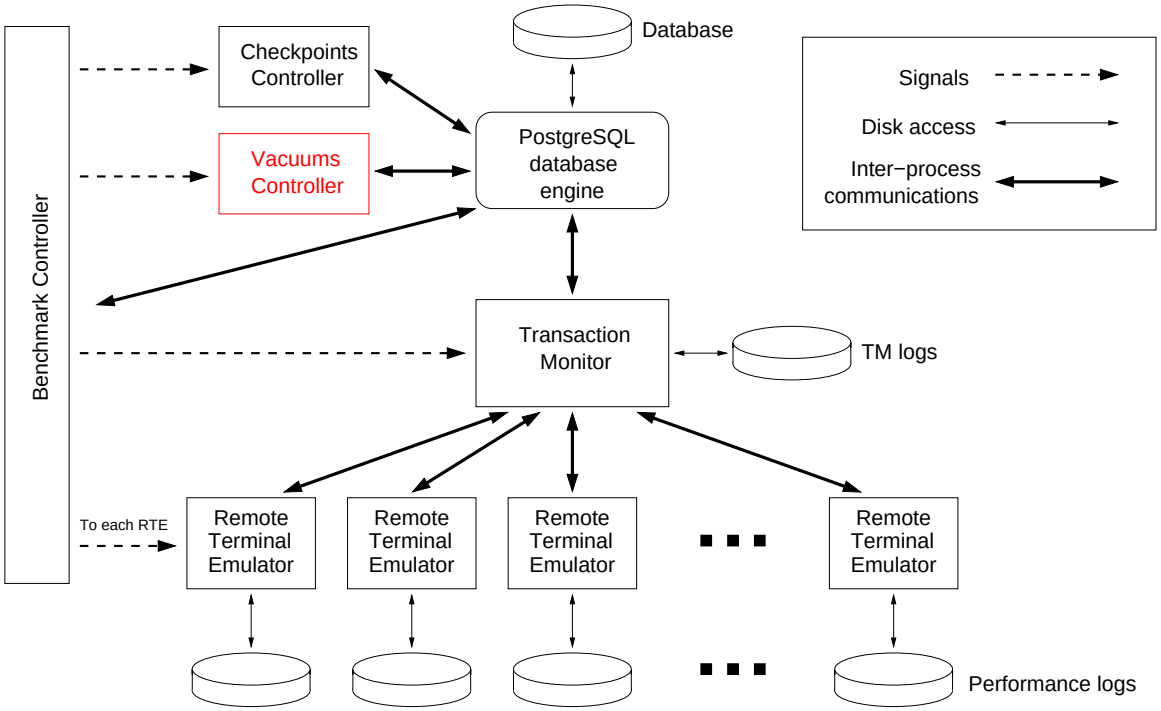
- The Transaction Monitor receives all the requests for RTEs and execute queries to the database system

TPCC-UVa architecture



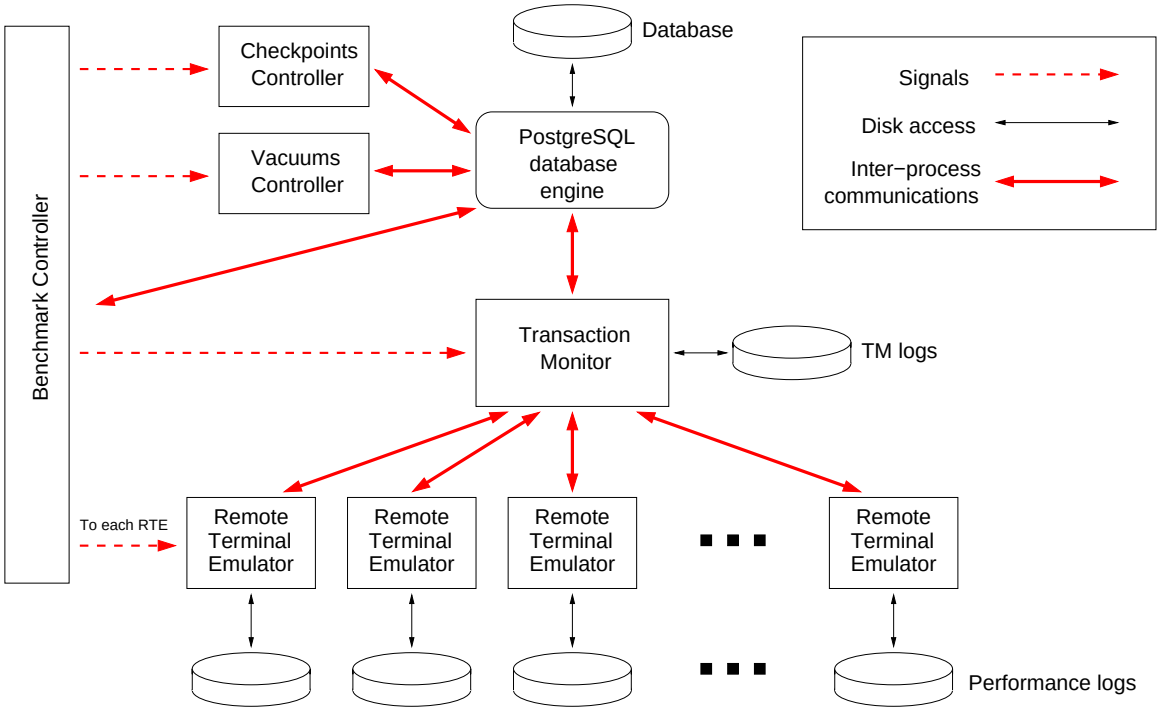
- The Checkpoints Controller performs checkpoints periodically and registers timestamps

TPCC-UVa architecture



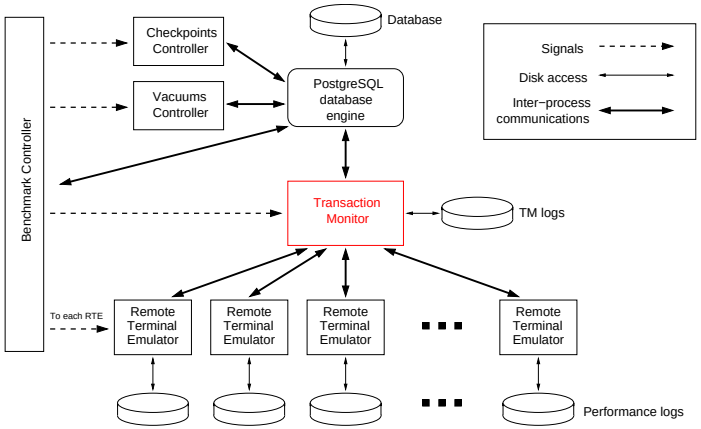
- The Vacuums Controller avoids the degradation produced by the continuous flow of operations in the database

TPCC-UVa architecture



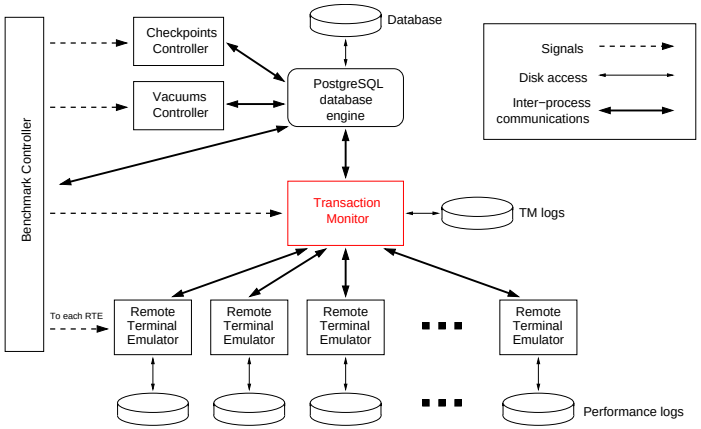
- IPCs are carried out using shared-memory structures and system signals → suitable for SMPs

The Transactions Monitor



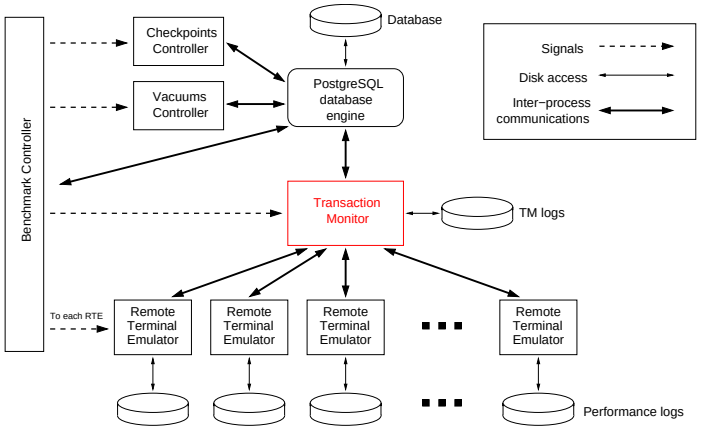
- The TM receives the transaction requests from all RTEs, passing them to the database engine and returning the results
- The TPC-C clause that forces the use of a “commercially available TM” avoids the use of tailored TMs to artificially increase performance
- We do not use a “commercially available TM”; instead, we simple queue the requests and pass them to the database

The Transactions Monitor



- The TM receives the transaction requests from all RTEs, passing them to the database engine and returning the results
- The TPC-C clause that forces the use of a “commercially available TM” avoids the use of tailored TMs to artificially increase performance
- We do not use a “commercially available TM”; instead, we simple queue the requests and pass them to the database

The Transactions Monitor



- The TM receives the transaction requests from all RTEs, passing them to the database engine and returning the results
- The TPC-C clause that forces the use of a “commercially available TM” avoids the use of tailored TMs to artificially increase performance
- We do not use a “commercially available TM”; instead, we simple queue the requests and pass them to the database

Running an experiment

- The TPC-C benchmark should be executed during a given period (2 or 8 hours), with a workload chosen by the user
- To be considered valid, the results of the test should meet some response time requirements (that is, the test may fail)
- Our implementation, TPCC-UVa, checks these requirements and reports the performance metrics, including tpmC-uva obtained
- Results given in the paper shows the performance of an Intel Xeon system with two processors, with a value for tpmC-uva $= 107.882$ for 9 warehouses

Running an experiment

- The TPC-C benchmark should be executed during a given period (2 or 8 hours), with a workload chosen by the user
- To be considered valid, the results of the test should meet some response time requirements (that is, the test may fail)
- Our implementation, TPCC-UVa, checks these requirements and reports the performance metrics, including tpmC-uva obtained
- Results given in the paper shows the performance of an Intel Xeon system with two processors, with a value for tpmC-uva = 107.882 for 9 warehouses

Running an experiment

- The TPC-C benchmark should be executed during a given period (2 or 8 hours), with a workload chosen by the user
- To be considered valid, the results of the test should meet some response time requirements (that is, the test may fail)
- Our implementation, TPCC-UVa, checks these requirements and reports the performance metrics, including tpmC-uva obtained
- Results given in the paper shows the performance of an Intel Xeon system with two processors, with a value for tpmC-uva = 107.882 for 9 warehouses

Running an experiment

- The TPC-C benchmark should be executed during a given period (2 or 8 hours), with a workload chosen by the user
- To be considered valid, the results of the test should meet some response time requirements (that is, the test may fail)
- Our implementation, TPCC-UVa, checks these requirements and reports the performance metrics, including tpmC-uva obtained
- Results given in the paper shows the performance of an Intel Xeon system with two processors, with a value for tpmC-uva $=$ 107.882 for 9 warehouses

Experimental results: Report

```
Test results accounting performed on 2004-18-10 at 17:58:57 using 9 warehouses.

 Start of measurement interval:  20.003233 m
 End of measurement interval:  140.004750 m
 COMPUTED THROUGHPUT: 107.882 tpmC-uva using 9 warehouses.
 29746 Transactions committed.

NEW-ORDER TRANSACTIONS:
 12946 Transactions within measurement time (15035 Total).
 Percentage:  43.522%
 Percentage of "well done" transactions:  90.854%
 Response time (min/med/max/90th):  0.006 / 2.140 / 27.930 / 4.760
 Percentage of rolled-back transactions:  1.012% .
 Average number of items per order:  9.871 .
 Percentage of remote items:  1.003% .
 Think time (min/avg/max):  0.000 / 12.052 / 120.000

PAYMENT TRANSACTIONS:
 12919 Transactions within measurement time (15042 Total).
 Percentage:  43.431%
 Percentage of "well done" transactions:  90.858%
 Response time (min/med/max/90th):  0.011 / 2.061 / 27.387 / 4.760
 Percentage of remote transactions:  14.862% .
 Percentage of customers selected by C_ID: 39.601% .
 Think time (min/avg/max):  0.000 / 12.043 / 120.000
```

Experimental results: Report

```
ORDER-STATUS TRANSACTIONS:
 1296 Transactions within measurement time (1509 Total).
 Percentage:  4.357%
 Percentage of "well done" transactions:  91.435%
 Response time (min/med/max/90th):  0.016 / 2.070 / 27.293 / 4.640
 Percentage of customers chosen by C_ID: 42.284% .
 Think time (min/avg/max):  0.000 / 9.998 / 76.000

DELIVERY TRANSACTIONS:
 1289 Transactions within measurement time (1502 Total).
 Percentage:  4.333%
 Percentage of "well done" transactions:  100.000%
 Response time (min/med/max/90th):  0.000 / 0.000 / 0.001 / 0.000
 Percentage of execution time < 80s :  100.000%
 Execution time min/avg/max:  0.241/2.623/27.359
 No.  of skipped districts:  0 .
 Percentage of skipped districts:  0.000%.
 Think time (min/avg/max):  0.000 / 4.991 / 38.000

STOCK-LEVEL TRANSACTIONS:
 1296 Transactions within measurement time (1506 Total).
 Percentage:  4.357%
 Percentage of "well done" transactions:  99.691%
 Response time (min/med/max/90th):  0.026 / 2.386 / 26.685 / 5.120
 Think time (min/avg/max):  0.000 / 5.014 / 38.000
```

Experimental results: Report

```
 Longest checkpoints:
 Start time Elapsed time (s) Execution time (s)
 Mon Oct 18 20:19:56 2004 8459.676000 27.581000
 Mon Oct 18 18:49:10 2004 3013.506000 21.514000
 Mon Oct 18 19:19:32 2004 4835.039000 14.397000
 Mon Oct 18 18:18:57 2004 1200.238000 13.251000

No vacuums executed.

» TEST PASSED
```

- If the test fails because of response time requirements have not met, the workload chosen was too high: The experiment should be repeated with less warehouses

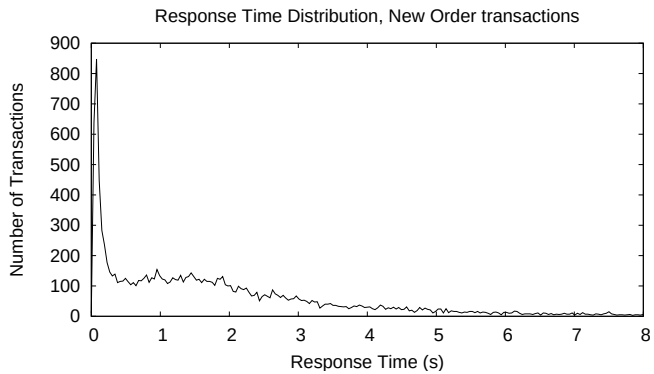
Experimental results: Report

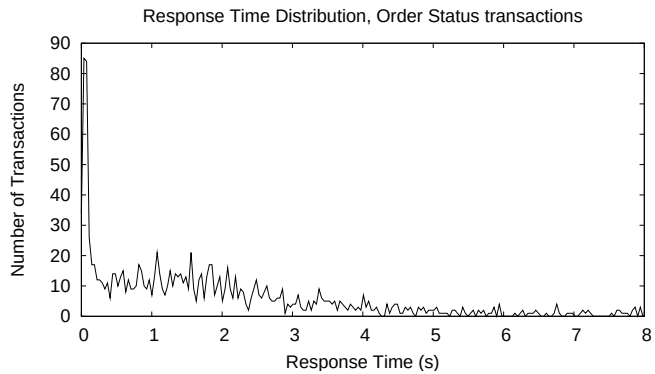
```
 Longest checkpoints:
 Start time Elapsed time (s) Execution time (s)
 Mon Oct 18 20:19:56 2004 8459.676000 27.581000
 Mon Oct 18 18:49:10 2004 3013.506000 21.514000
 Mon Oct 18 19:19:32 2004 4835.039000 14.397000
 Mon Oct 18 18:18:57 2004 1200.238000 13.251000

No vacuums executed.

» TEST PASSED
```

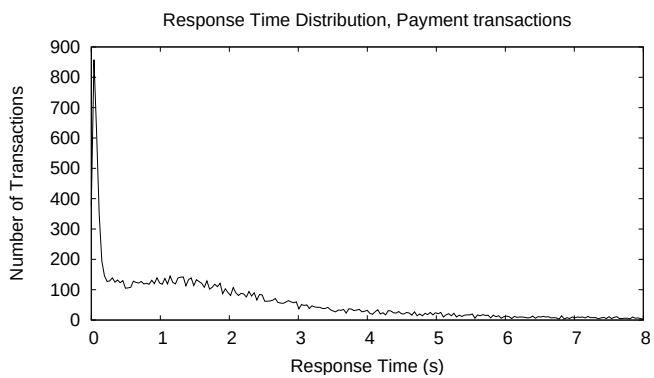
- If the test fails because of response time requirements have not met, the workload chosen was too high: The experiment should be repeated with less warehouses

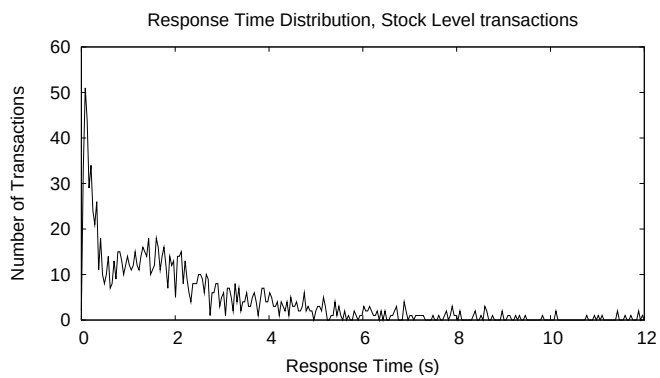
Experimental results: Plots

- According with clause 5.6.1 of TPC-C, some performance plots should be generated after a test run





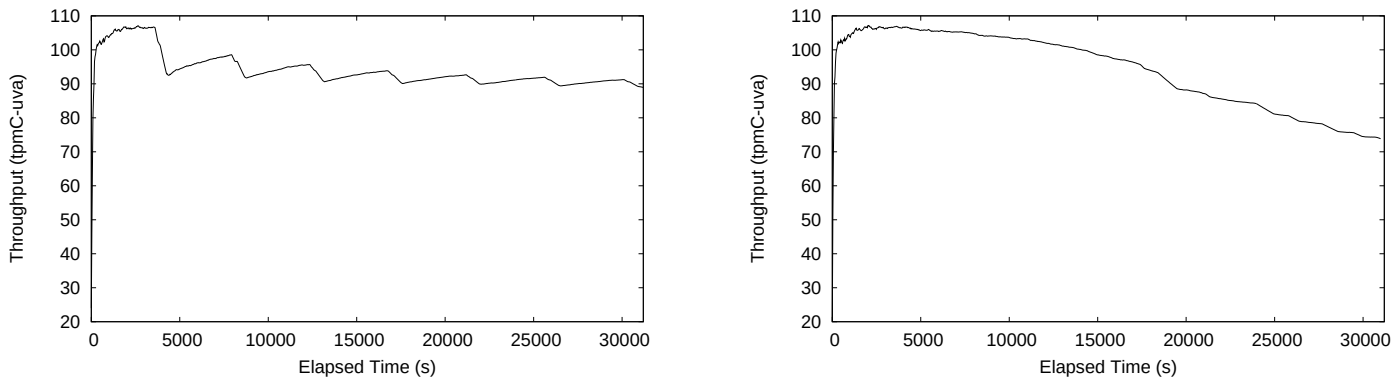




Response time distribution of some transaction types for a 2-hours execution on the system under test

Experimental results: Need of vacuums

- If the experiment is longer than 8 hours, vacuums should be executed periodically in order to keep performance



Throughput of the New-Order transaction for a 2-hours execution on the system under test With (a) hourly vacuum operations, and (b) no vacuums.

Conclusion

- TPCC-UVa is an implementation of TPC-C benchmark that allows the performance measurement of parallel and distributed systems
- TPCC-UVa is open-source, making easy to instrument it in order to use it with simulation environments such as Simics
- TPCC-UVa can be downloaded from

http://www.infor.uva.es/~diego/tpcc-uva.html

Conclusion

- TPCC-UVa is an implementation of TPC-C benchmark that allows the performance measurement of parallel and distributed systems
- TPCC-UVa is open-source, making easy to instrument it in order to use it with simulation environments such as Simics
- TPCC-UVa can be downloaded from

http://www.infor.uva.es/~diego/tpcc-uva.html

Conclusion

- TPCC-UVa is an implementation of TPC-C benchmark that allows the performance measurement of parallel and distributed systems
- TPCC-UVa is open-source, making easy to instrument it in order to use it with simulation environments such as Simics
- TPCC-UVa can be downloaded from

http://www.infor.uva.es/~diego/tpcc-uva.html

TPCC-UVa

An Open-Source TPC-C Implementation for Parallel and Distributed Systems

Diego R. Llanos, Belén Palop

Universidad de Valladolid

Computer Science Department
University of Valladolid, Spain

PMEO-PDS 06, Rhodes Island, April 29th, 2006